DSC 440, HW2

Kefu Zhu

2.3 Suppose that the values for a given set of data are grouped into intervals. The intervals and corresponding frequencies are as follows:

age	*frequency*
1–5	200
6–15	450
16–20	300
21–50	1500
51–80	700
81–110	44

Compute an *approximate median* value for the data.

Answer:

Based on the formula

$$median = L_1 + (\frac{N/2-(\sum freq)_l}{freq_{median}}) \cdot width$$

The approximate median value is $21 + (\frac{3192/2-(200+450+300)}{1500}) \cdot 30 \approx 33.51$

2.6 Given two objects represented by the tuples (22, 1, 42, 10) and (20, 0, 36, 8):

(a) Compute the *Euclidean distance* between the two objects.

(b) Compute the *Manhattan distance* between the two objects.

(c) Compute the *Minkowski distance* between the two objects, using $q = 3$.

(d) Compute the *supremum distance* between the two objects.

Answer:

(a)

$$Euclidean\ distance = \sqrt{(22-20)^2 + (1-0)^2 + (42-36)^2 + (10-8)^2} \approx 6.7082$$

(b)

$$Manhattan\ distance = 2 + 1 + 6 + 2 = 11$$

(c)

$$Minkowski\ distance\ (q=3) = (|22-20|^3 + |1-0|^3 + |42-36|^3 + |10-8|^3)^{1/3} \approx 6.1534$$

(d)

$$supremum\ distance = \max_f^p |x_{if} - x_{jf}| = 6$$

2.7 The *median* is one of the most important holistic measures in data analysis. Propose several methods for median approximation. Analyze their respective complexity under different parameter settings and decide to what extent the real value can be approximated. Moreover, suggest a heuristic strategy to balance between accuracy and complexity and then apply it to all methods you have given.

Answer:

I propose and compare two differeent methods for median approximation here: **median of the medians** and **estimate by interpolation**

- **median of the medians**:

  Breaking the entire list to sublists of 5 items will take $O(n)$. Sort each sublist and determine the median for each sublist. Since every sublist is short (only have 5 items), it takes $O(n)$. We then reecursively determine the median of the set of medians generated from sublists. The total complexity of this approach on average is $O(n)$

- **estimate by interpolation**:

  $$median = L_1 + \left(\frac{N/2 - (\sum freq)_l}{freq_{median}}\right) \cdot width$$

  To create the equal-width bins, we need to sort the data first, which on averege has complexity of $O(n\ logn)$. Putting items into the correct bin takes $O(n)$, which can also be done along the sorting process. Extracting the median bin and count the number of items in there takes $O(width)$. Computing the number of items in bins that are lower than the median bin takes $O(n)$. Since

$O(width) < O(n) < O(n\ logn)$, the total complexity is $O(n\ logn)$.

To gain more efficiency, rather than dividing into sublists of 5 items, we could divide the entire list into somewhat larger sublists (e.g. 11 items) with fewer number of sublists in total. That way we can do less recursive operations later on in the sacrifice of some accuracy.

Similarly, since the complexity of **estimate by interpolation** comes mainly from sorting process, we can adapt the concept of diving entire list into sublists idea. That will result in faster sorting time overall and also loses some accuracy on the other hand.

2.8 It is important to define or select similarity measures in data analysis. However, there is no commonly accepted subjective similarity measure. Results can vary depending on the similarity measures used. Nonetheless, seemingly different similarity measures may be equivalent after some transformation.

Suppose we have the following 2-D data set:

	A_1	A_2
x_1	1.5	1.7
x_2	2	1.9
x_3	1.6	1.8
x_4	1.2	1.5
x_5	1.5	1.0

(a) Consider the data as 2-D data points. Given a new data point, $x = (1.4, 1.6)$ as a query, rank the database points based on similarity with the query using Euclidean distance, Manhattan distance, supremum distance, and cosine similarity.

(b) Normalize the data set to make the norm of each data point equal to 1. Use Euclidean distance on the transformed data to rank the data points.

Answer:

(a)

	A_1	A_2	**Euclidean**	**Manhattan**	**supremum**	**cosine similarity**
x_1	1.5	1.7	1	1	1	1
x_2	2	1.9	5	5	2	4
x_3	1.6	1.8	3	3	4	2
x_4	1.2	1.5	2	2	4	3
x_5	1.5	1.0	4	4	2	5

(b)

After normalization, the new data point, $x = (0.4667, 0.5333)$

	A_1	A_2	**Euclidean**
x_1	0.4688	0.5312	1
x_2	0.5128	0.4872	4
x_3	0.4706	0.5294	2
x_4	0.4444	0.5556	3
x_5	0.6	0.4	5

Note: The ranking based on normalized data is the same with cosine similarity now

3.1 *Data quality* can be assessed in terms of several issues, including accuracy, completeness, and consistency. For each of the above three issues, discuss how data quality assessment can depend on the *intended use* of the data, giving examples. Propose two other dimensions of data quality.

Answer:

- **Accuracy**:

  Different users of the data may have different definition for what is "accurate". A data scientist may consider a dataset as inaccurate if 20% of the data are outdated or missing, while a business analyst may not think of it the same way since with the rest 80% accurate data, he can already draw an accurate conclusion of the overall market situation.

- **Completeness**:

  Facing the same dataset, users with different intent will have different feedback on the completeness of the data. For some users, maybe they do not need some of the features in the dataset that are missing to conduct their anaylsis and they feel the dataset is complete (They never realize some features are missing since they only work with a portion of the dataset).

  On the other hand, some other people will have completely opposite feedback about the completeness of the data if they work with the entire dataset or need information from features that have missing values.

- **Consistency**:

  Different users may have their own way on working with the same feature in a dataset. For people who work with transformed values such as percentage, the inconsistent use of unit have zero influence on them, which will have huge impact on users who work directly with the raw value.

Some other dimentions that can be used to measure the quality of data include **timeliness** and **interpretability**.

3.3 Exercise 2.2 gave the following data (in increasing order) for the attribute *age*: 13, 15, 16, 16, 19, 20, 20, 21, 22, 22, 25, 25, 25, 25, 30, 33, 33, 35, 35, 35, 35, 36, 40, 45, 46, 52, 70.

(a) Use *smoothing by bin means* to smooth these data, using a bin depth of 3. Illustrate your steps. Comment on the effect of this technique for the given data.

(b) How might you determine *outliers* in the data?

(c) What other methods are there for *data smoothing*?

Answer:

(a):

Steps:

1. Group every 3 data points together in a bin
2. Replace each value in a bin with the mean value of the bin

Applying the smoothing by bin means can remove some noise in the data

(b):

Outliers exist if any bins (usually the first or the last bin) have extremely lower/higher bin values compared to their neighbors

(c): We could also smooth the data by fitting some reegressions or removing outliers by clustering

3.5 What are the value ranges of the following *normalization methods*?

(a) min-max normalization

(b) z-score normalization

(c) z-score normalization using the mean absolute deviation instead of standard deviation

(d) normalization by decimal scaling

Answer:

(a) min-max normalization: $[-\infty, +\infty]$, the range can be defined by users based on different intents

(b) z-score normalization: $[-\infty, +\infty]$

(c) z-score normalization using the mean absolute deviation: $[-\infty, +\infty]$

(d) normalization by decimal scaling: $[-1.0, 1.0]$

3.7 Using the data for *age* given in Exercise 3.3, answer the following:

(a) Use min-max normalization to transform the value 35 for *age* onto the range [0.0, 1.0].

(b) Use z-score normalization to transform the value 35 for *age*, where the standard deviation of *age* is 12.94 years.

(c) Use normalization by decimal scaling to transform the value 35 for *age*.

(d) Comment on which method you would prefer to use for the given data, giving reasons as to why.

Answer:

(a): min=max normalization

$$v_i' = \frac{v_i - \min(A)}{max(A) - min(A)}(new\_max_A - new\_min_A) + new\_min_A = \frac{35-13}{70-13} * 1 + 0 \approx 0.3860$$

(b): z-score normalization

$$v_i' = \frac{v_i - \bar{A}}{\sigma_A} = \frac{35-29.96}{12.94} \approx 0.3893$$

(c): Normalize value 35 by decimal scaling will results in 0.35

(d): I would prefer using the z-score normalization because it does not suffer from "out-of-bound" problem like min-max normalization does, and also preserves the original distribution of data better than the decimal scaling

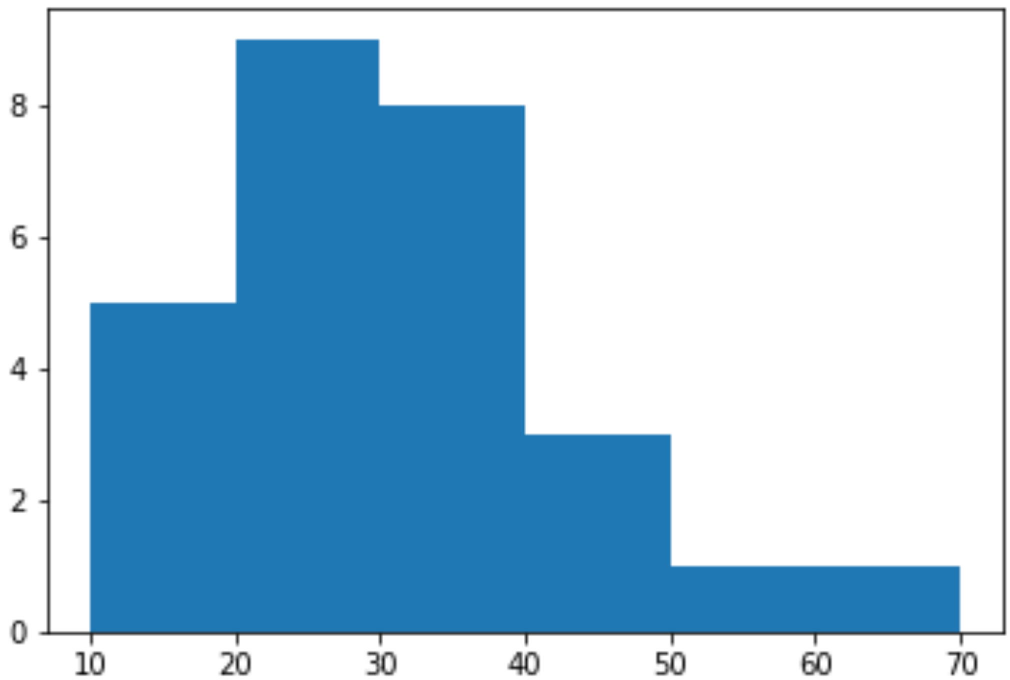
3.11 Using the data for *age* given in Exercise 3.3,

(a) Plot an equal-width histogram of width 10.

(b) Sketch examples of each of the following sampling techniques: SRSWOR, SRSWR, cluster sampling, and stratified sampling. Use samples of size 5 and the strata "youth," "middle-aged," and "senior."

Answer:

(a)



(b)

T1	13	T6	20	T11	25	T16	33	T21	35	T26	52
T2	15	T7	20	T12	25	T17	33	T22	36	T27	70
T3	16	T8	21	T13	25	T18	35	T23	40		
T4	16	T9	22	T14	25	T19	35	T24	45		
T5	19	T10	22	T15	30	T20	35	T25	46		

- SRSWOR: [40, 35, 70, 25, 36]
- SRSWR: [22, 16, 40, 16, 30]

- Cluster sampling: Each column can be considered a cluster, we pick s = 2

T1	13
T2	15
T3	16
T4	16
T5	19

T16	33
T17	33
T18	35
T19	35
T20	35

- Stratified sampling:

T1	13	young	T6	20	young	T11	25	young	T16	33	Middle-aged	T21	35	Middle-aged	T26	52	senior
T2	15	young	T7	20	young	T12	25	young	T17	33	Middle-aged	T22	36	Middle-aged	T27	70	senior
T3	16	young	T8	21	young	T13	25	young	T18	35	Middle-aged	T23	40	Middle-aged			
T4	16	young	T9	22	young	T14	25	young	T19	35	Middle-aged	T24	45	Middle-aged			
T5	19	young	T10	22	young	T15	30	Middle-aged	T20	35	Middle-aged	T25	46	Middle-aged			

T2	15	young
T7	20	young
T15	30	Middle-aged
T22	36	Middle-aged
T26	52	senior

3.13 Propose an algorithm, in pseudocode or in your favorite programming language, for the following:

(a) The automatic generation of a concept hierarchy for nominal data based on the number of distinct values of attributes in the given schema.

(b) The automatic generation of a concept hierarchy for numeric data based on the *equal-width* partitioning rule.

Answer:

(a)

```
begin

// Initialize a dictionary to store, Attribute:Unique_Counts
concept_hierarchy = {}

// Loop over all attributes and
// count number of unique values within each attribute
for each attribute 'X' in schema:
    'X'_unique = count distinct 'X' values
    concept_hierarchy['X'] = 'X'_unique

// Rank the hierarchy based on number of unique values of attributes
sort concept_hierarchy by values in ascending order

end
```

(b)

```
begin

// Initialize a dictionary to store, Attribute:Unique_Counts
concept_hierarchy = {}

// For each level of hierarchy (different attributes)
for each attribute 'X' in schema:
    // User defined width
    width_'X' = USER INPUT
    // User defined min and max for binning
    _min_'X' = USER INPUT
    _max_'X' = USER INPUT
    // Compute the number of bins (steps) for each level of hierarchy
    steps_'X' = (_max_'X' - _min_'X')/width_'X'

// First level of hierarchy (X1: first attribute)
for i from 1 to steps_'X1':
    // 'Current' key stores the values for each bin
    // in the first level of hierarachy
    concept_hierarchy[i] = {'Current': []}
    // Second level of hierarchy (X2: second attribute)
    for j from 1 to steps_'X2':
        // 'Current' key stores the values for each bin
        // in the first level of hierarachy
        concept_hierarchy[i][j] = {'Current': []}
```

```
        // More for loops if have more levels of hierarchy
        for ...

// 1. Rank data based on first level of hierarchy

// Initialize the min and max bound for binning
cur_min_'X1' = _min_'X1'
cur_max_'X1' = cur_min_'X1' + width_'X1'
// Iterate over all bins
for i from 1 to steps_'X1':
    // For every value in the first attribute
    for m in 'X1':
        # If the value falls into the current bin
        if m >= cur_min_'X1' and m < cur_max_'X1':
            # Put the data into the bin
            concept_hierarchy[i]['Current'] = m

    // One step forward (Change bin)
    cur_min_'X1' = cur_max_'X1'
    cur_max_'X1' = cur_min_'X1' + width_'X1'

// 2. For each data point in the first level of hierarchy
//     rank data based on second level of hierarchy

// For each bin of first level hierarchy
for i from 1 to steps_'X1':
    // Find their values for second attribute
    'X2'_sub

    // Set min and max bound binning in level 2 hierarchy
    cur_min_'X2' = _min_'X2'
    cur_max_'X2' = cur_min_'X2' + width_'X2'
    // Iterate over all bins in level 2 hierarchy
    for j from 1 to steps_'X2':
        // For every value in the second attribute
        for n in 'X2'_sub:
            # If the value falls into the current bin
            if n >= cur_min_'X2' and n < cur_max_'X2':
                # Put the data into the bin
                concept_hierarchy[i][j]['Current'] = n

        // One step forward (Change bin)
        cur_min_'X2' = cur_max_'X2'
        cur_max_'X2' = cur_min_'X2' + width_'X2'
```

```
// More loops if have more levels of hierarchy
...
```